International Journal of Technology Enhanced Learning (2011), 3 (1), 80-92.
DOI: http://dx.doi.org/10.1504/IJTEL.2011.039065

User Identity Issues in Mashups for Learning Experiences using IMS Learning Design

Luis de la Fuente Valentín, Derick Leony, Abelardo Pardo, Carlos Delgado Kloos
Department of Telematics Engineering
Carlos III University of Madrid, Spain

April 29, 2010

Abstract

The combination of services that provide personal information in technologies such as educational mashups brings some issues in the management of users' identity and authorization. This article presents a scenario based on the fact that an IMS LD server requires information relevant to each learner, and this information is provided by external services. This scenario allows to describe the problems of user correspondence, authenticated data retrieval, and remote account creation; a solution using technologies currently available is provided for each, as well as recommendations to take into account in similar scenarios.

1 Introduction

The Web 2.0 is characterized by an enormous number of tools with usage ranging from purely leisure activities to the most dedicated work. Because of such a wide variety of options, there are a large number of possibilities for these tools to be used with educational purposes. This educational application is straightforward in the case of communication tools such as forums or video-conference rooms [7]. In other cases, the usefulness of the application is highly dependent on the context, like the case of simulations [4] or serious games [10]. But aside from these issues, the use of Web 2.0 tools in education environments resulted in the appearing of the so called pedagogy 2.0 [9].

The ideal scenario would be for technology to be so flexible as to provide a learning environment that can be adjusted and configured with the best available tools for each desired functionality. The concept of *mashup* is build upon this idea: different applications are combined or "mashed up" into a single one. On its first approximation, a mashup consists on the aggregation of a customized set of tools into a common web space with the easy access as the prominent added value [11]. The next step is the combination of these functionalities to create a new tool, the features of which cannot be obtained from the other tools in isolation [16]. This idea is not exclusive of the learning scenarios, it is also used in others contexts such as news (BFreeNews [1]), weather forecast (Woozor [2]), etc.

In e-Learning systems, resources and services in activities are typically orchestrated in such a way that the completion of the activities produces the desired learning. This is the

[1] `http://www.bfreenews.com`
[2] `http://woozor.com`

goal of the instructional design, the discipline of designing effective learning experiences. Formalisms such as IMS Learning Design [8] (henceforth IMS LD) allow for the definition and capture of the structure of a learning experience in a form suitable to be interpreted as many times as required. In the context of instructional design, we use the term *mashup* as a collection of web tools provided by different vendors (referred in this article as third party services) and orchestrated by a workflow engine.

IMS LD, the workflow engine in use for the purpose of this paper, does not natively provide mechanisms to orchestrate third party services. There are different proposals to overcome this drawback. For example, [15] use the concept of widgets to integrate third party tools in IMS LD courses. The approach used in this paper is the Generic Service Integration (GSI) [12] architecture, which allows bidirectional information exchange among the workflow engine and the external tool.

The scenario described in this document assumes that there is a central computational server where the course flow is being enacted and obtains student data from third party services. This structure presents several challenges with respect to privacy and authentication: how can the server in charge of the orchestration act on behalf of course participants without affecting their privacy and with no disruption of the learning flow?

The aim of this paper is to analyze and discuss user identity and authorization issues that arise when designing the interaction among services from different parties in an educational context. To support the analysis, we describe a scenario where IMS LD is used to orchestrate the use of third party services and the rest of the course material. The solutions presented are based on the use of the OpenID [1] and OAuth [2] specifications. This study is part of the implementation of external service integration in Grail, the IMS LD run-time environment in .LRN. The described solutions have been implemented and tested in such system [5].

The rest of the paper is organized as follows. Section 2 describes in detail the mashup scenario discussed in the document. In Section 3, specific problems are analyzed and their corresponding solutions are exposed. Then, a practical implementation of the solutions is presented in Section 4. Finally, conclusions are included Section 5.

2 Definition of the Mashup Scenario

The analysis performed in this article is based upon a mashup constructed for educational purposes. The scenario description is done generically: the text states about *third party services*, each of them could be any Web 2.0 available tool. Not all these services have been created to be used in e-Learning, but the overall composition of them provide a course flow with complete pedagogical meaning.

It is worth to note that the term *mashup* usually implies a decoupled and loose integration among services. The aggregation proposed by GSI requires a configuration step that suggests a stronger interaction. The course flow is therefore conducted by an orchestrator of resources and services, which is feed by one or more external services in two different ways.

- In one hand, activities are sequenced by following a set of case specific established rules. Services are used in the context of different activities. The course flow decides which service accompanies each activity.
- On the other hand, the course flow, or the learning material itself, can be adapted depending on students' interactions with the material. In the mashup scenario, third party service's data sources the course adaptation.

In the reciprocal way, the orchestrator provides contextual information to the external services so they are aware of the background in which they are being used and can provide more personalized information.

IMS LD has been chosen as the course flow orchestrator. It consists of an open specification that provides support to represent a wide range of pedagogical strategies. Thus, it offers a hierarchical vocabulary of learning terms: *activities* are grouped within *structures*, which are placed into *acts*. *Acts* form part of a *play* and there can be several different plays in one same pedagogical *method*. The definition of *roles* also provides support for collaborative learning strategies. The underlying complexity of IMS LD is divided into three levels: A is the basic and core level; B includes the properties and conditions; and C incorporates notifications. The representation of the pedagogical model is transcribed into a script and is packaged together with the resources needed to deploy it; this package is called Unit of Learning (or simply UoL). Any UoL can be deployed in any compliant runtime environment, with the same behaviour in all of them.

Authoring tools, applications that assist authors in the creation of a UoL, are referred to as Learning Design editor, while tools that assist learning staff in the enactment of the UoL are usually called Learning Design runtime environments or players. GRAIL belongs to the latter category, although it provides some edition capabilities [6]. The IMS LD specification does not define a way to interact with external services, an analysis in depth of this problem and a proposed solution for it are presented in [12].

The mashup scenario is then composed by two clearly differentiated sides. First, the IMS LD server that performs the orchestration of the different elements. The IMS LD server compiles all the information and processes it to offer the next activity, according to a pedagogical plan. The second part is the collection of third party services, Web 2.0 tools whose composition offer new possibilities to the pedagogical plan.

There are two ways of using external services in the described scenario: human of programmatic.

The former consists in the interaction of a human being with the service GUI. Students access the service interface though a link that contextualized an activity. The goal is to produce learning with student's interaction. There is no discussion about how students can access their own account in the external service: once they click the proper link, they access to the interface and they are prompted for authentication, unless the browser already have a *cookie* that maintains a previous session. Thus, each user enters into her space in third party service.

The latter type, programmatic access, consists in the communication among the IMS LD server and the third party one. This interaction is started by the orchestrator and usually relies on the use of a previously agreed protocol, such as SOAP, REST or a proprietary protocol. In any case, there is no human intervention in the process. However, the system needs the retrieval of private user's data, which implies that the user have previously shared this information with the system

The programmatic interaction with third party services presents several security and privacy issues that hinders the use of such scenario in real situations. In the next section, these problems are listed and analyzed in detail, and practical standard-based solutions are presented for them.

3 Analysis of the Problem

The previously described scenario shows the potential richness of mashups in the context of IMS LD scripted courses: the integration of third party services in learning flows allow the inclusion of Web 2.0 tools (and therefore Web 2.0 based pedagogical models) into e-learning material. By accessing external tools, course participants can perform learning activities with domain specific tools. Furthermore, if data about the user interaction is obtained from the third party service, the following material in the learning scenario can be adapted.

Programmatic access to external services is usually implemented using public Application Programming Interface (API) offered by the third party service provided. This type of access does not require user's intervention and does not generate any user session. However, user's data retrieval may be affected by privacy policies restricting the access to data unless the explicit approval by the user is obtained. This fact poses a severe restriction on the described scenario, which can be summarized in the following question: How can the service act on behalf of course participants without affecting their privacy and with no disruption on the learning flow?

This section analyzes the problem of external user management. The proposed solution decomposes the problem into smaller parts where a simpler solution can be applied.

3.1 Correspondence of Identities

If the mashup used in a course needs external service's data to adapt its appearance or content, the IMS LD server must pragmatically retrieve and store this data. Furthermore, there are cases in which the retrieved data is not related to a given user, but to a more generic environment. For example, the number of times a given resource has been used, or the resource with the highest ranking according to participant's votes. The opposite case appears when the data belongs to a single user, for instance, the last comment posted from the user in the external service.

The latter presents a challenge to the IMS LD server because the user already has a unique identifier in the local server (e.g. john@LDserver.com). But, assuming he has already been registered in the external service, which is the identifier that corresponds to the same person? As shown in Figure 1, the local server needs to find the correspondence between these two identities, so that it is able to retrieve the remote user data and relate it to the proper local user. It may happen that the same person used the same email address to register in both servers. However, it is not necessarily the case, so a procedure to find correspondence of identities needs to be defined.

Again, the ideal solution where the entire process is automatic requires metadata that associates the accounts in both servers. However, this is not the common case. It follows a discussion of the possible procedures to solve the problem.

The straightforward approach is to prompt the user for her username in the external service. Despite its simplicity, this solution presents some drawbacks:

- Each service included in the mashup introduces one additional activity in the course flow (establish this correspondence). This addition of activities may slow down the course itself, and could result in a negative effect from the point of view of the cognitive load [13].
- Course cannot start until all participants have indicated their external username. It means that if a user does not perform the activity, all the scheduled activities are de-

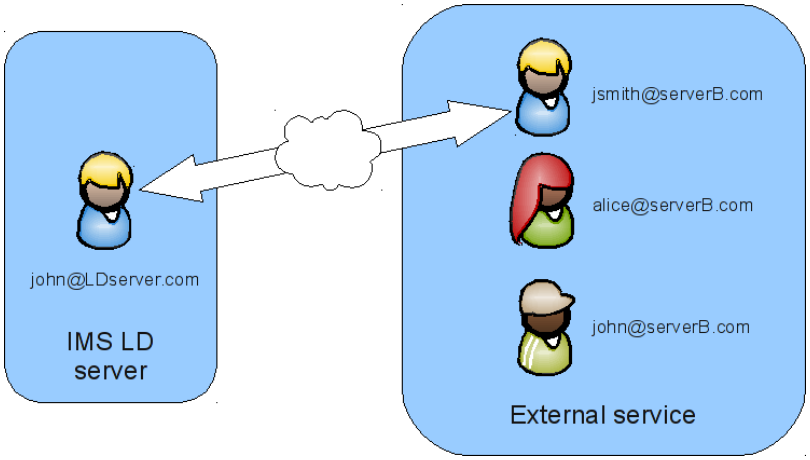

Figure 1: IMS LD server does not know the user identity in the third party server.

layed. Course flow could include alternatives to avoid the problem, but its complexity increases as the number of students grows and hinders the authoring phase.

OpenID is an authentication system that allows a unique, distributed identity to be used on any compliant server [1]. A person who has an OpenID account can use it as registration information in different services, instead of having several usernames and passwords. This system can be used to find correspondence between users in the mashup scenario: if both the LMS and the external service provide authentication with OpenID, it is guaranteed that the same identifier corresponds to the same person.

However, this solution also presents some drawbacks, namely:

- There is no restriction in the number of OpenID accounts held by the same person. It is possible for the same person to have different identifiers in different servers. A manual step to confirm correspondence is still needed but it is much less restrictive than in the previous case, because no user action can be considered as the acceptance of the correspondence and does not block the course flow.
- The catalogue of web tools that have adopted OpenID restricts the number of services to be included in a mashup.

The third solution, based on the use of OAuth [2], provides a way to find this correspondence and authorize the access user data both at the same time. This solution is discussed in the following section.

3.2 Authentication for Data Retrieval

A service that offers data through the public API must ensure that the other party owns the corresponding privileges. Typically, the owner of the data (the student) must actively grant access to her data.

As in the previous problem, the IMS LD server could prompt students for external service's login and, additionally, password. The privacy problem is clear here: there is no reason for users to reveal their password to anybody. In the hypothetical case in which

the user trust the IMS LD server and facilitates her data, security problems such as man-in-the-middle attacks [3] rely on eavesdropping an insecure channel and if this takes place during the authentication phase, the attacker is able to gain access to the user credentials. As a result, storing the user credentials for the external site in the IMS LD server is not recommended either because it compromises these credentials.

The most suitable solution is to use the authentication protocol known as OAuth [2]. The use of such protocol requires the following steps:

- The requester agent, in this case the IMS LD server, initiates the OAuth negotiation with the external service. As a response, the external service provides a *request token* to the IMS LD server. The external service will also provide a token secret to be used together with the request token.
- The IMS LD server then redirects the user's browser to a previously set URL, hosted by the third party service. If the user has not started a session in the service yet, there will be a redirection to a sign-in form; and after the user introduces the proper credentials, a request to allow access to the IMS LD server is presented.
- When the user agrees to allow the IMS LD server to access the information stored in the external service, the IMS LD server receives an *access token*, which differs from the first one in that it is an authenticated token, and it can be used as proof of authentication to retrieve data from the external server.
- In the next interactions between the IMS LD server and the external service, the IMS LD server uses both the access and the secret tokens. The external service is then able to validate if the tokens are correct and if they belong to the owner of the requested data. If these conditions are met, then the external service serves the requested information.

A graphical representation of these steps is shown in Figure 2. In this example, the IMS LD server wants to retrieve information belonging to user Bob, which is provided by external service G. In the worst case, Bob would have to provide the credentials he uses in service G to the IMS LD server; instead of this, service G offers an API with authentication performed through OAuth. Therefore, Bob allows service G to provide his information to the IMS LD server through the authentication OAuth token. This token, stored and used by the IMS LD server, is implicitly related to Bob's identity on the third party service. That is, the correspondence between identities has also been established, implicitly solving the problem previously stated.

One of the disadvantages of OAuth is the lack of granularity for privileges within its permission system, since the only capability provided to the user is either to grant or not all privileges to the IMS LD server. This low granularity implies that even if the IMS LD server needs only read privilege to a small portion of information, it will be granted full permissions, depending on the actions provided through the API by the external service.

Another drawback of OAuth is the constant lifetime of the authentication token. The lack of flexibility regarding the lifetime of the token affects in a special way those courses with duration longer than the period of time in which the token will be available. In this scenario, the learners would see their UoL interrupted at some point, since the communication between the IMS LD server and the external service would need the token to be updated in order to continue with the validation.

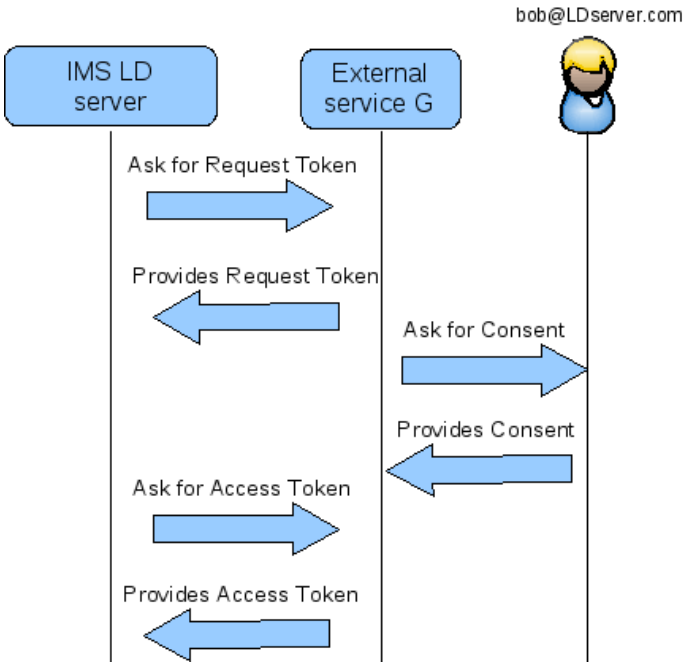

Figure 2: Exchange of messages between IMS LD server and external service to authenticate user Bob.

3.3 Automatic User Creation

As discussed in subsection 3.1, there is a need to find a correspondence of user accounts between different services. This correspondence issue translates into a more specific problem, which is the automatic creation of a user account in the external service.

The previously presented problems rely on the assumption that the user has previously created an account within the external service. Considering the amount of possible learners involved in the UoL and the large number of external services available, it is very unlikely that every learner will have an account in all the used external services.

Current anti-spam technologies, such as "captchas" [14], precludes the possibility of creating user accounts automatically in the external service. Thus, a different approach is required.

In a scenario where all services are controlled by the same authentication entity, such as corporative platforms, this issue is not present. When using technologies such as LDAP or ActiveDirectory, one credential allows access to all the services. However, this centralization is what makes this solution less feasible for open and decentralized scenarios.

Consequently, a mashup opened to the Web 2.0 is assumed to require user's intervention during the creation of their accounts. The goal is to minimize time consumption during this process. The most feasible solution is the use of OpenID to delegate the management of user accounts to a third-party service. An identity managed by an OpenID provider is complemented by user's data, so that they can be used to create the desired user account.

If both the IMS LD and the third party service provide OpenID support, the OpenID identifier can be directly used in the creation of the account. The third party service asks

the OpenID provider for additional user's data. The process finishes when the user herself accepts the creation of the new account.

However, this solution presents difficulties when used in an educational mashup environment:

- Although the external server uses OpenID to authenticate users, it may require an additional registration to match the OpenID with an account. As a consequence, although the process for signing up is eased by the use of OpenID, the user needs to deviate slightly from the instructional flow described in the UoL in order to register within the external service.
- Some services may require more information than the one provided by the OpenID provider, it is usually not possible to provide this information in an automatic way through an API, therefore this has to be taken into account when choosing what external service to use with the IMS LD server. Passing the fulfilment of this critical information to the learner might bring problems during the execution of the UoL and to the flow of the whole course in general.

4 Practical implementation

The mashup scenario described in section 2 has been implemented as part of GRAIL, and has been deployed in practical situations. This section explains how the solution to the discussed problems can be effectively used in the IMS LD runtime environment.

The first problem appears with the IMS LD specification itself, because it does not provide a precise construction to include third party services into Units of Learning. The services included in the specification (conference, send-mail, monitor, index) are not enough to get all the pedagogical potential that can be captured by Web 2.0 tools when used as part of the course flow. To overcome this problem, GRAIL includes the "Generic Service Integration" [12] (hence GSI), a proposed extension for IMS LD that allows for the inclusion of third party services in the context of pedagogical flows conducted by Learning Design.

This model covers the whole life-cycle of a course. To use a service in a UoL, its functionality must be described with the vocabulary offered by GSI. When the course is deployed in a compliant platform, the provided description allows to choose the actual tool that will be used in the course, taken from a catalogue of available services. The mapping among users from GRAIL and the external service is performed during the configuration of the course instance, when all other resources are being allocated. This working scheme allow the creation of service mashups in the context of educational courses, where IMS LD coordinates the use of the different services and uses their information to orchestrate the complete course flow.

During the runtime, GSI acts as a middle-layer that mediates in the communication among the runtime environment and external services. Thus, events occurring in the IMS LD side can have a direct effect on the external service. For example: the termination of an activity may result in configuration changes in the third party tool. Web 2.0 tools can usually be accessed by a public API, which is case specific. Supporting the inclusion of a service means that GSI is able to use its API. The architecture of GSI is build on service specific modules, independent among themselves, each of them providing support for a given third party tool. Then, a service is said to be available for its use in a course if the correspondent module has been loaded in the runtime environment.

Each third party service defines its own privacy policies. This implies that some services offer OpenID or OAuth support, and some others do not. Therefore, the actual practical

solution for the problems discussed in section 3 will be different on each module. As a consequence, some services require human intervention during the course instance configuration, but some other services do not. For example: a service hosted in the same LMS as the IMS LD runtime environment does not require any access granted to the data, while an OAuth supporting service from a third party provider requires the intervention of the user to explicitly allow the IMS LD runtime to access the external data.

The intervention of course participants during service configuration presented a problem in GRAIL. The configuration interface is only available to platform administrators, while course participants can only interact with the course instance when all resources and services have been allocated. The goal then was to allow non-privileged users to participate in course configuration without letting them access the administrator interface nor creating a new specific user interface, which would decrease the tool usability.

The adopted solution was to include the so called *zero act*, which is a set of activities that are presented to the user as if they were part of the course activities, but they do not belong to the original UoL. If required, a GSI service can include one or more of these activities in the *zero act*. These activities are similar to other regular activities (they contain an activity and an environment), but they are not part of the pedagogical flow, they are used to configure the required external services.

The actual course begins when the *zero act* finishes. That is, when all participants have performed the requested task, or when the administrator forces its completion. Thus, course beginning is not blocked by users that did not finish the configuration activities.

As a proof of concept of the theoretical discussion stated on this paper, we have implemented a GSI module that includes support for assessment by means of Google Spreadsheets [5]: the answers to a questionnaire given by students are stored in a Spreadsheet, which is hosted by Google. Then, typical spreadsheet functions can be applied to grade the answers. In a final step, GRAIL retrieves the calculated values and stores it as an IMS LD property, which will be used as input for course content adaptation.

The Google Docs public API offers OAuth capabilities for data retrieval. Thus, the *zero act* consists of an activity (see Figure 3) where the user is prompted to allow GRAIL access to their Google Docs data by simply clicking on a link. The resulting token is user specific, so it maps the local user account with the external one and at the same time grants access to the external data.

5 Conclusions

This paper analyses a mashup scenario where services are orchestrated by an IMS Learning Definition of a course flow. Learning content can be adapted to user profiles, preferences or abilities, by considering their activity in the course. It includes their interactions with the services that form the mashup. Thus, the IMS LD server needs to retrieve students' data that is hosted by third parties. The identity and authentication issues that appear in the described mashup are:

- Assuming that the user already has an account in the third party service, how the IMS LD engine knows which account corresponds to each user? OpenID provides unique identities that can be used to log on different web applications. If both servers support it and the same identifier exists on them, it is guaranteed that they belong to the same user.
- Assuming the correspondence of identities among servers has been correctly established, how can the IMS LD engine retrieve third party data without affecting users

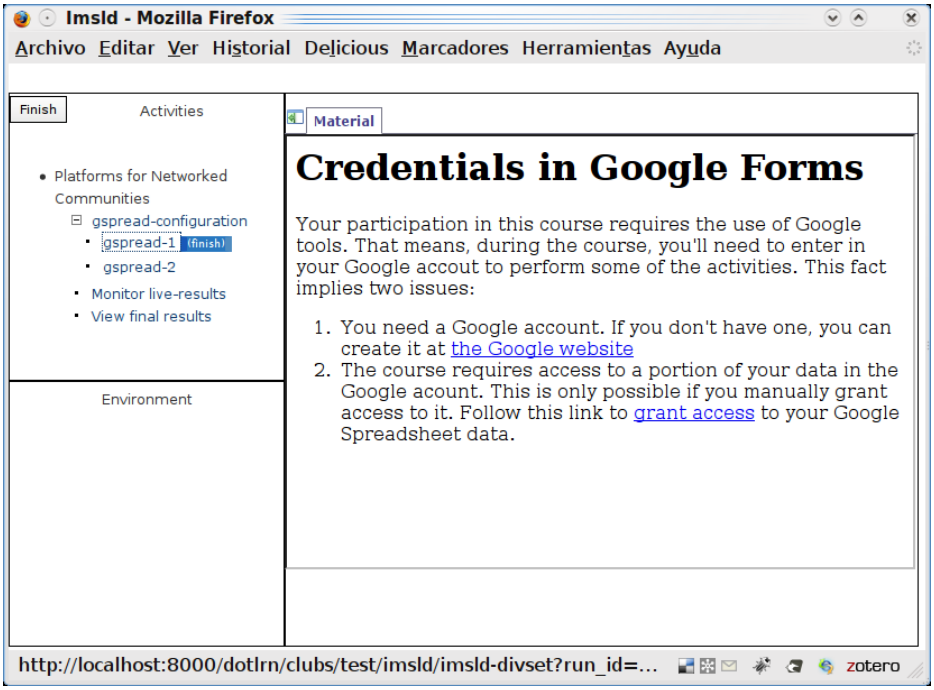

Figure 3: Zero Act activity for Google Spreadsheet GSI module.

privacy? If the third party service supports OAuth, it can generate a token that allow its owner to access user's data without revealing her identity.

- The usage of a third party service requires that every student have already an account on it. OpenID eases the process of account creation, still needing the user intervention.

As discussed in the paper, the actual solution depends on the features supported by the service in use. Thus, each implementation of a service integration would require manual intervention in a different degree, with the overall goal of haven a complete automated process.

The discussions presented in the paper have a practical implementation based on GRAIL, an IMS LD runtime environment integrated with the .LRN LMS. In such tool, Learning Design has been enhanced with Generic Service Integration, which allows the inclusion of external services in a course flow conducted by IMS LD. The exposed example, based on Google Spreadsheet as external service, revealed that the inclusion of an additional step is useful to perform the required configuration activities. This step has been called *zero act* and is automatically created by the runtime engine when external services are in the course flow. The course begins when services configuration - that is, the *zero act* - has finished, so that this act does not interfere with the actual learning flow.

The analyzed scenario combines the computational richness of mashups with the pedagogical expressiveness of instructional design. Resulting courses can therefore exploit all the pedagogical potential of the so called Web 2.0.

Acknowledgment

This work has been partially funded by the Project Learn3 (TIN2008-05163/TSI) from the Plan Nacional I+D+I, the Spanish National Project FLEXO (TSI-020301-2008-19, www.ines.org.es/flexo) and "Investigación y Desarrollo de Tecnologías para el e-Learning en la Comunidad de Madrid" funded by the Madrid Regional Government under grant No. S2009/TIC-1650.

References

[1] OpenID specification. `http://openid.net/specs.bml`, 2006.

[2] OAuth core 1.0. http://oauth.net/core/1.0/, December 2007. Last visited April 2009.

[3] N. Asokan, Valtteri Niemi, and Kaisa Nyberg. Man-in-the-Middle in tunnelled authentication protocols. In *Security Protocols*, pages 28–41. 2005.

[4] S. de Freitas and M. Oliver. How can exploratory learning with games and simulations within the curriculum be most effectively evaluated? *Computers & Education*, 46(3):249–264, 2006.

[5] Luis de la Fuente Valentín, Abelardo Pardo Sánchez, and Carlos Delgado Kloos. Using third party services to adapt learning material: A case study with google forms. In *Learning in the Synergy of Multiple Disciplines*, pages 744–750, Niza, October 2009. Springer.

[6] J. Escobedo del Cid, L. de la Fuente Valentín, S. Gutiérrez, A. Pardo, and C. Delgado Kloos. Implementation of a Learning Design Run-Time Environment for the .LRN Learning Management System. Journal of Interactive Media in Education, 2007.

[7] B. Giesbers, B. Rienties, W.H. Gijselaers, M. Segers, and D.T. Tempelaar. Social presence, Web videoconferencing and learning in virtual teams. *Industry and Higher Education*, 23(4):301–309, 2009.

[8] IMS Learning Design specification. http://www.imsglobal.org/learningdesign/, Feb. 2003. Last visited April 2009.

[9] Catherine McLoughlin and Mark J. W. Lee. Future learning landscapes: Transforming pedagogy through social software. *Journal of Online Education*, 4(5), 2008.

[10] P. Moreno-Ger, D. Burgos, I. Mart\'ınez-Ortiz, J. L Sierra, and B. Fernández-Manjón. Educational game design for online education. *Computers in Human Behavior*, 24(6):2530–2540, 2008.

[11] D.E. Simmen, M. Altinel, V. Markl, S. Padmanabhan, and A. Singh. Damia: data mashups for intranet applications. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1171–1182. ACM, 2008.

[12] Luis Fuente Valentin, Yongwu Miao, Abelardo Pardo, and Carlos Delgado Kloos. A supporting architecture for generic service integration in IMS learning design. In *Proceedings of the 3rd European conference on Technology Enhanced Learning: Times of Convergence: Technologies Across Learning Contexts*, pages 467–473, Maastricht, The Netherlands, 2008. Springer-Verlag.

[13] Jeroen J. G. van Merriënboer and John Sweller. Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review*, 17(2):147–177, June 2005.

[14] Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford. CAPTCHA: using hard AI problems for security. In *Advances in Cryptology — EUROCRYPT 2003*, page 646. 2003.

[15] S. Wilson, P. Sharples, and D. Griffiths. Extending IMS Learning Design services using Widgets: Initial findings and proposed architecture. In *Proc. of the 3rd TENCompetence Open Workshop on Current Research on IMS Learning Design and Lifelong Competence Development Infrastructures*, 2007.

[16] J. Wood, J. Dykes, A. Slingsby, and K. Clarke. Interactive visual exploration of a large spatio-temporal dataset: Reflections on a geovisualization mashup. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1176, 2007.